API 설명 문서

db = SQLAlchemy(app)

@app.route('/patient/total')

def P\_T():

result = db.session.query(func.count('\*')).select\_from(Patient).scalar()

print(result)

return render\_template('patient/total.html', result=result)

@는 데코레이터를 말하는것이며 함수를 감싸는 형태로 되어있고 주로 기존 함수를 수정하지 않으면서 추가 기능을 구현할 때 사용한다.

app.route(‘/patient/total’) 는 플라스크로 실행해서 선언한 app의 주소가 /patient/total 일 때 그 밑에 구문의 함수가 실행을 하게 되어짐

db.session.query(func.count('\*')).select\_from(Patient).scalar()

db는 위의 코드에 사전으로 세팅되어 할당한 데이터베이스를 db변수에 저장한것임

session은 세션을 활성화하여 db의 쿼리문을 실행하게 만들음

.query는 SQL에서 조회문에서의 Select와 From 사이에 어느 속성을 조회할 것인지를 말함

Func.count(‘\*’)는 SQL에서의 count(\*)와 같이 해당 테이블의 카디널리티를 구해줌

Select\_from(Patient)는 SQL구문에서의 FROM 절과 같다. 이 구문은 사전에 만들어둔 데이터 모델중 Patient 모델을 사용하여 데이터를 조회하는 것임

Scalar()는 해당 쿼리를 실행한후 첫 번째 결과의 첫 번째 요소를 반환하거나 행이 없으면 None을 반환함

그리고 쿼리를 실행함으로써 나온 리스트 캆을 result 변수에 저장을 한다.

render\_template('patient/total.html', result=result)는 실행된 후 나온 결과 리스트 값인 result를 result라는 변수에다 저장한다. 그리고 'patient/total.html' 해당 경로에 있는 파일에다 result 값을 전송한 후 'patient/total.html' 파일에 전송 받은 값을 파싱하여 리스트 값을 출력하면 된다.